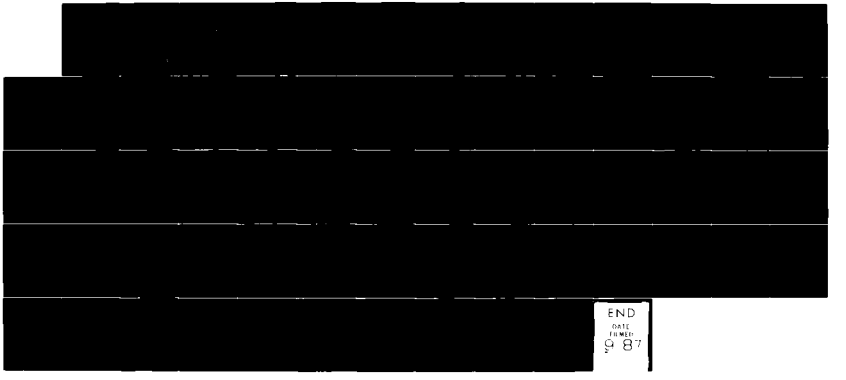
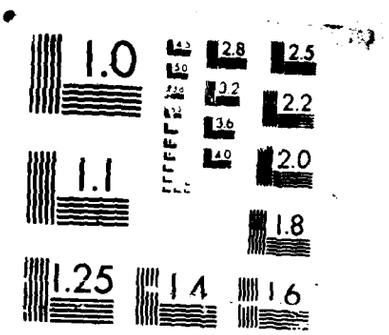
AD-A183 797 AN EMPIRICAL STUDY OF PLAN-BASED REPRESENTATIONS OF PASCAL AND FORTRAN CO..(U) RUTGERS - THE STATE UNIV NEW BRUNSWICK NJ COGNITION AND COMPU.. 1/1

UNCLASSIFIED S P ROBERTSON ET AL. JUN 87 CCL-0687-001 F/G 12/5 NL



END
DATE FILMED
9 87



MICROCOPY RESOLUTION TEST CHART

COGNITION
AND
COMPUTING
LABORATORY
Report No. CCL-0687-001

AN EMPIRICAL STUDY OF PLAN-BASED REPRESENTATIONS OF PASCAL AND FORTRAN CODE

Scott P. Robertson
Chiung-Chen Yu

Department of Psychology
Rutgers University
Busch Campus
New Brunswick, NJ 08903

June 1987

Sponsored by:

Perceptual Science Programs
(Code 1142PS)
Office of Naval Research
Contract No. N00014-86-K-0876
Work Unit No. NR 4424203-01

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CCL-0687-001	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): An Empirical Study of Plan-Based Representations of Pascal and Fortran Code		5. TYPE OF REPORT & PERIOD COVERED: Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Scott P. Robertson, Chiung-Chen Yu		8. CONTRACT OR GRANT NUMBER(s): N00014-86-K-0678
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Cognition and Computing Laboratory, Psychology Dept.,Rutgers Univ.-Busch Campus, New Brunswick, NJ 08903		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR 4424203-01
11. CONTROLLING OFFICE NAME AND ADDRESS: Perceptual Science Programs, Office of Naval Research, Arlington, VA 22217		12. REPORT DATE: June, 1987 13. NUMBER OF PAGES: 54
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report): Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approvrd for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software psychology, human computer interaction, program comprehension planning.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The first step in program modification is comprehension. Several researchers have argued recently that programmers utilize a plan-based representation when composing or comprehending program code. In a series of studies we are testing the psychological validity of this proposal and examining the nature of plan-based program representations. Here we report the results of our first study in which programmers segmented code and sorted programs. The segmenting data showed

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

that programmers agree on the major components of a program and that these components are defined by goals in a plan representation. Pascal and Fortran programs that employ the same plan structures were segmented into similar components. These components were labelled with similar subgoal descriptions. The majority of subgoals described were abstract, with few being task-specific. Program sorting data also shows clustering into plan groups, however some secondary dimensions, like the type of data used (in Pascal programs), may also be important parts of program representation.

Introduction

How is computer program code conceptually represented, and how do programmers utilize conceptual representations of code in the task of program modification? In a series of experiments, we have set out to explore these questions using a variety of psychological methodologies. Here the first of these experiments is reported.

Several researchers have argued recently that programmers utilize a plan-based representation when composing or comprehending program code (Adelson, 1981; Bonar & Soloway, 1985; Soloway, Bonar & Erlich, 1983; Soloway & Erlich, 1984; Soloway, Ehrlich, Bonar & Greenspan, 1982; Ehrlich & Soloway, 1983; Rist, 1986). Soloway and his colleagues are at the forefront of theory in this area. They have developed a taxonomy of programming plans and plan types. For example, Bonar & Soloway (1985) note that novice programmers bring "step by step knowledge" about how to make choices, iterate, and perform other sequential activities from everyday life into the programming task. They must then acquire "programming knowledge," which consists not only of the syntax and semantics of a programming language, but also the *plans*, or hierarchically organized sequences of goals and actions that achieve specific tasks.

Some empirical studies of the claim that programmers utilize plans have appeared. Rist (1986), for example, asked novice and expert programmers to group lines of Pascal code that "did the same things" together. Novice and expert

programmers grouped lines on several crtiteria including "global plans" like *initialize, input, calculate,* and *output,* "task-level" plans like *sorting wallpaper costs for a specific room* or *calculating the tax for a particular item,* and "syntax", like *assignment statements* or *loop control statements*. Interestingly, novices grouped many more lines based on syntax while experts grouped lines based on function.

In this experiment, we explored the way in which expert programmers chunk code by looking at several different programs, including programs in different languages, that utilize the same plans. Previous researchers have tended to focus on plans in individual programs, assuming that the abstract plans are transferable across programs. Also, previous research has almost exclusively dealt with a single language, usually Pascal. If we truly believe that plans are abstract knowledge structures that programmers utilize when they write or read code, then we should be able to demonstrate their common properties across programs and across languages.

Plans are knowledge structures that organize steps in a procedure into chunks. Each chunk achives a subgoal in the goal hierarchy of a particular task. Program plans organize sections of code into chunks. Task-level plans consist of subgoals that are specified in the task language, to "get an address from a buyer's list" for example. General plans, on the other hand, consist of subgoals that are abstractions

from a specific task, and may in fact apply in several contexts, "iterate in a loop" for example. Programs that perform different tasks but use the same general plans should be chunked the same way. To examine this claim, we asked programmers to segment code from Pascal and Fortran programs and to sort the programs into groups. Within each language group, there were subgroups of programs that utilized the same general plans. A major goal of our initial study was to demonstrate that plan subgroups are recognizable to programmers by showing that programs sharing a plan would be sorted together and by showing that programs sharing a plan would be segmented and described the same way.

A programming language is usually designed to support general plans. Thus, languages contain "loop constructs" but do not contain "compare phone lists" constructs. To the extent that general program plans are abstract from specific tasks, programs that are written in different languages but that utilize the same plan should be recognized as similar. A second goal of this study was to show that Pascal and Fortran programs which used the same plan would be segmented and labelled in the same way.

Method

Subjects. Fifteen Pascal programmers and fifteen Fortran programmers were recruited from the student population at Rutgers University. Most of the Pascal programmers were

graduate students in computer science while most of the Fortran programmers were engineering graduate students. Each subject was paid $8.00 per hour for participation, and most subjects spent 1.5-2.5 hours in the experiment.

Materials. Nine Fortran programs and nine Pascal programs were written for use in this (and subsequent) experiments. All of the programs are debugged, working code. The nine programs in each language group achieved nine different tasks, but they were written in accordance with the three plans shown in Table 1. Within each language group, there were three programs that utilized each of the three plans.

Pascal programs Pas1, Pas2, and Pas3 and Fortran programs For1, For2, and For3 were written in accordance with the first plan in Table 1, the "MGOM" plan. The MGOM plan consisted of five subgoals: 1) declare data structures 2) display a MENU, 3) GET an input from the menu, 4) perform the OPERATION selected by the input, and 5) return to the MENU state or quit. Programs Pas1 and For1 were data analysis programs, Programs Pas2 and For2 were computer mail programs, and programs Pas3 and For3 emulated an electronic calculator. The texts of these programs appear in Appendix A.

Pascal programs Pas4, Pas5, and Pas6 and Fortran programs For4, For5, and For6 were all written in accordance with the second plan in Table 1, the "RCP" plan. The RCP plan consisted of four subgoals: 1) declare data structures,

2) READ lists from files, 3) COMPARE lists and get common elements, and 4) PRINT the common elements. Programs Pas4 and For4 found common courses in transcripts and printed these as transfer courses, programs Pas5 and For5 compared two schedules and printed possible meeting times, and programs Pas6 and For6 compared two mailing lists and printed common customers. The texts of these programs appear in Appendix B.

Pascal programs Pas7, Pas8, and Pas9 and Fortran programs For7, For8, and For9 were written in accordance with the third plan in Table 1, the "RTRDP" plan. The RTRDP plan consisted of five subgoals: 1) declare data structures, 2) READ and TEST an initial input, 3) display further information and READ new inputs, 4) DO a transaction or calculation, and 5) PRINT the results. Programs Pas7 and For7 emulated a bank teller machine, programs Pas8 and For8 presented a stimulus and collected a reaction time as if for a psychology experiment, and programs Pas9 and For9 controlled a computer login sequence.

Procedure. Subjects were run individually or in small groups. Each subject received a packet containing instructions and the nine programs in either Pascal or Fortran. Subjects were first instructed to draw lines between statements in the code in order to "identify the parts" of the program and to divide each program into "several major sections." Each time a subject drew a segment

Table 1: Components of the three programming plans.

Plan 1: "MGOM"

a. declare data structures.

b. display a MENU.

c. GET an input from the menu.

d. perform the selected OPERATION.

e. return to the MENU state or quit.

Instantiations of plan 1

Data analysis (For1, Pas1).

Computer mail (For2, Pas2).

Electronic calculator (For3, Pas3).

Plan 2: "RCP"

a. declare data structures.

b. READ lists from files.

c. COMPARE lists and get common elements.

d. PRINT the common elements.

Instantiations of plan 2

Course transfer (For4, Pas4).

Schedules (For5, Pas5).

Mailing lists (For6, Pas6).

Table 1. (cont.)

Plan 3: "RTRDP"

a. declare data structures.

b. READ and TEST an initial input.

c. display further information and READ new input.

d. DO a transaction or calculation.

e. PRINT the results.

Instantiations of plan 3

Computer login sequence (For7, Pas7).

Stimulus-response psychology experiment (For8, Pas8).

Bank teller (For9, Pas9).

line it was numbered in order. After segmenting the major sections of a program the subjects were instructed to write a descriptive label for each major section "in terms of the program's task." For each program, after segmenting and labelling the major sections, the subjects were instructed to draw segment lines within the major sections to identify subsections. Finally, subjects were asked to sort the programs into groups. The subjects were told that programs belonged in a group if they "work the same way." Subjects were allowed to form as many groups as they wished but they could not leave a single program in a group by itself nor could they put all of the programs together into a single group.

Results

Subject Programming Experience. Subjects in both the Pascal and Fortran groups had an average of 3.6 years of programming experience. However, subjects in the Pascal group reported having worked with more programming languages (a mean of 5.8 languages) than subjects in the Fortran group (a mean of 2.8 languages), $t(27)=4.59$, $p<.001$.

Segmenting. Each subject drew line segments in the code of nine programs. We predicted that line segments would be drawn at plan boundaries. For each program, we calculated the frequency of line segments drawn after each line. If 60% or more of the subjects segmented a program at a

Table 2. Frequencies of segmenting by 60% or more of the subjects at predicted and not predicted positions in the Pascal programs.

	Segmenting predicted		Segmenting not predicted	
	Observed	Not Observed	Observed	Not Observed
Pas1	4	0	0	53
Pas2	4	0	0	46
Pas3	3	1	0	48
Pas4	3	1	0	46
Pas5	3	1	0	76
Pas6	2	2	0	31
Pas7	3	1	0	44
Pas8	4	0	0	43
Pas9	3	1	0	71
Pascal Totals	29	7	0	458

Table 3. Frequencies of segmenting by 60% or more of the subjects at predicted and not predicted positions in the Fortran programs.

	Segmenting predicted		Segmenting not predicted	
	Observed	Not Observed	Observed	Not Observed
For1	3	1	0	46
For2	3	1	0	33
For3	3	1	0	45
For4	4	0	0	53
For5	4	0	2	51
For6	4	0	0	45
For7	3	1	0	46
For8	4	0	0	32
For9	3	1	1	52
Fortran Totals	31	5	3	403

particular line, we considered that line an important chunk boundary. Important chunk boundaries should correspond to the predicted plan boundaries.

Tables 2 and 3 show the frequencies of important chunk boundaries (those that were segmented by 60% or more of the subjects) at predicted and not-predicted locations for the nine Pascal programs and the nine Fortran programs respectively. Chi-squares on the frequencies for all programs were significant, ranging from $x2(1)=24$, $p<.05$ for Pas6 to $x2(1)=78$, $p<.001$ for Pas5. Twenty-nine out of 36 (81%) of the plan boundaries in the Pascal programs were segmented according to our criterion. Thirty-one out of 36 (86%) of the plan boundaries in the Fortran programs were segmented according to our criterion. None of the 458 non-boundary lines were segmented in the Pascal programs, and only 3 out of 406 (<1%) non-boundary lines were segmented in the Fortran programs.

Modal Labels of Program Segments. After segmenting the programs, the subjects labelled each section with a description of its function. We were concerned with whether these descriptions corresponded with the subgoals that we claim control each chunk. Tables 4,5, and 6 show the *modal* descriptive labels that subjects gave to each chunk of each program in the MGOM, RCP, and RTRDP plan groups respectively. Included in this list are labels that were given to major chunks, those which were identified by a

Table 4. Modal descriptive labels for plan components in Pascal and Fortran programs using the MGOM plan.

Data analysis

Pas1	For1
Declare.	Declare.
Print instruction.	Display menu.
Get number.	Read key.
Compute means.	Calculate mean and update result.

Computer Mail

Pas2	For2
Declare variable.	Declare.
Print instruction.	Print our menu.
Process message.	Get input and print message.
Quit.	Check if end.

Electronic Calculator

Pas3	For3
Declare.	Declare.
Print Menu.	Print out menu.
Read in value and allow choice.	---
Do calculation.	Calculate.
---	Continue or end.

Table 5. Modal descriptive labels for plan components in Pascal and Fortran programs using the RCP plan.

Course transfer

Pas4	For4
Declare.	Define variables.
Read file.	Read data file.
If same then transfer.	Compare data.
Output results.	Print results.

Schedules

Pas5	For5
Declare.	Declare.
Read file.	Read files.
Compare lists.	Compare.
Print result.	Print out.

Mailing Lists

Pas6	For6
Declare.	Declare.
Read data.	Read data file.
Compare lists.	Compare data.
Print result.	Print out.

Table 6. Modal descriptive labels for plan components in Pascal and Fortran programs using the RTRDP plan.

Bank teller machine

Pas7	For7
Declare.	Declare.
Check password.	Read data.
Display options.	Print menu.
Print account.	Decrement account and print result.

Psychology experiment

Pas8	For8
Declare.	Declare.
Initialize variable.	Give instruction.
Print out message and receive response.	Get response and update count.
Calculate percentage and print result.	Output result.

Computer login sequence

Pas9	For9
Declare.	Declare.
Print account.	Check input.
Do transaction.	Choose.
Print result.	Print result.

segment line drawn by 60% or more of the subjects. Each label listed is the most frequent of the set of labels given to that chunk (the experimenters judged paraphrases and close matches in wording to be the same label).

In glancing over these lists, note that descriptions of the Pascal and Fortran programs which did exactly the same thing (e.g. Pas1 and For1, Pas2 and For2, Pas3 and For3, etc.) sound very similar. Also, the six programs within a plan group also sound similar.

Almost all of the labels are abstract, describing general computational functions such as "declare variables," "read data file," "display menu," "calculate," and "print results." Only a few labels are task specific, namely "compute means" and "calculate means" in Pas1 and For1 respectively, "if same then transfer" in Pas4, "check password" in Pas7, "decrement account" in For7, "calculate percentage" in Pas8, and "print account" in Pas9.

Descriptions of the plan chunks. Table 7 shows the major chunks of each plan and indicates whether or not 60% or more of the subjects provided an appropriate descriptive label for each chunk in each of the six programs. Subjects provided appropriate descriptions for all of the chunks in all of the instances of the RCP plan. This was the most successful set of programs in terms of matching label data to a plan.

Table 7. Production of appropriate decriptions for plan components. "Yes" indicates that 60% or more of the subjects provided an appropriate description.

MGOM Plan	Pas1	Pas2	Pas3	For1	For2	For3
Declare data structure.	yes	yes	yes	yes	yes	yes
Display menu/instruct.	yes	yes	yes	yes	yes	yes
Get/read input.	yes	no	yes	yes	yes	yes
Calculate/compute.	yes	yes	yes	yes	no	no
Quit.1	no	yes	no	no	yes	yes

RCP Plan	Pas4	Pas5	Pas6	For4	For5	For6
Declare data structure.	yes	yes	yes	yes	yes	yes
Read files.	yes	yes	yes	yes	yes	yes
Compare lists.	yes	yes	yes	yes	yes	yes
Print results.	yes	yes	yes	yes	yes	yes

RTRDP Plan	Pas7	Pas8	Pas9	For7	For8	For9
Declare data structure.	yes	yes	yes	yes	yes	yes
Test/initialize inputs.	yes	yes	no	yes	no	yes
Display instr./read inp.	yes	yes	yes	yes	yes	no
Do transaction.	no	yes2	yes	yes2	yes	yes
Print results.	yes	yes2	yes	yes2	yes	yes

1*"Quit" was an unexpected label, see text for a discussion.*

2*These components were described together in one label.*

For the MGOM plan, a majority of the subjects provided descriptions for the first two chunks, "Declare data structure" and "Display Menu/Instructions," in each program. Fewer than our 60% criterion provided a label for the "Get/Read Input" chunk in Pas2, but this chunk was included in descriptions of all the other programs. The final chunk, "Calculate/Compute", was included in descriptions of all the Pascal programs and For1, but not For2 and For3. Apparently many subjects felt that getting input and performing a computation were part of the same chunk in these latter two programs. Finally, subjects included an unexpected chunk, which they labelled "Quit," in Pas2, For2 and For3. This chunk is branched to when the "Quit" option is chosen in the "Get Input" part of the program.

For the RTRDP plan, descriptions were provided by a majority of subjects for all of the plan chunks in Pas8 and For7. In Pas9 and For8 the second chunk, "Test/Initialize Inputs" did not meet the 60% criterion and in those cases the chunk was included as part of the "Display Instructions/Read Input" subgoal. In For9, the "Display Instructions/Read Input" and "Test/Initialize Inputs" chunks were also combined, but described as "Check/Initialize Inputs." Finally, "Do Transaction" and "Print Results" were described as separate chunks in Pas9, For8 and For9, but were described togther in Pas8 and For7. "Do Transaction" did not meet the 60% criterion in Pas7.

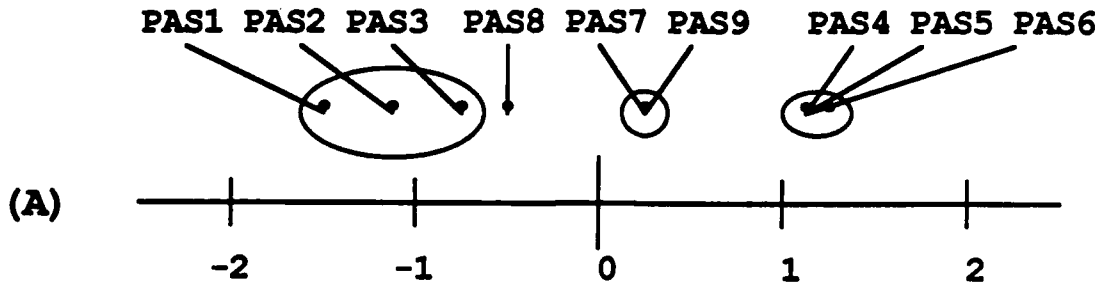
Table 8. Stress values for 1-3 dimensional solutions to multidimensional scaling of Pascal and Fortran program sorting data.

	Programming Language	
Dimensionality	Pascal	Fortran
One	.196	.124
Two	.002	.000
Three	.000	.000

Program Sorting. After segmenting and labelling, the programmers were asked to sort the programs into groups according to "the way they work." They were instructed to sort the programs into as many groups as they wished with the constraints that no program could be left by itself and all the programs must not be sorted together into a single group.

Multi-dimensional scaling (MDS) was used to determine if the Pascal and Fortran programs were sorted into three distinct plan groups. For each language group, the input to the MDS algorithm was a matrix of the frequencies with which each program was sorted with each of the other programs. Use of the frequency data is based on the assumption that programs which are more similar will be sorted together more frequently. Separate MDS analyses were performed on the Pascal and the Fortran data. The strongest prediction is that a one-dimensional solution will fit both data sets well and that the plot of the stimulus coordinates will show three clusters based on the plan groups.

Table 8 shows the "stress" values for one, two, and three-dimensional MDS solutions for both the Pascal and the Fortran data. Lower stress values indicate a good fit to the data, and values below .15 are considered to be good fits (Kruskal & Wish, 1978). On these criteria, the one dimensional solution fits the Fortran data very well and the Pascal data fairly well. A two dimensional solution



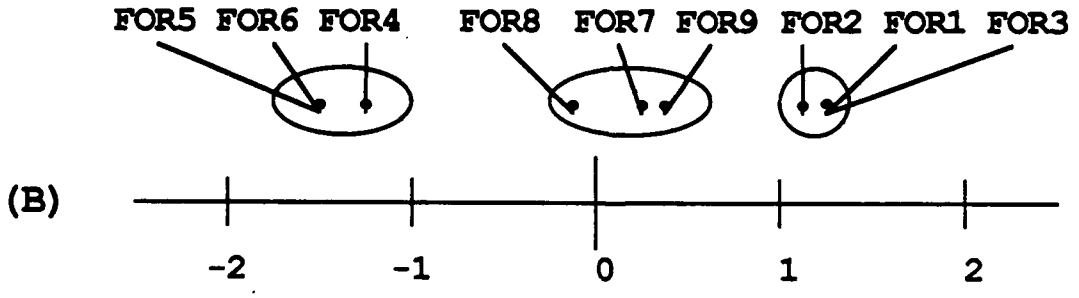


Figure 1. Plots of the stimulus coordinates for one-dimensional solutions to MDS of Pascal (A) and Fortran (B) program sorting data.

completely explains the distributions of data for both the Fortran programs (although Kruskal & Wish, 1978 suggest that a one-dimensional solution with stress below .15 is adequate) and the Pascal programs.

Figure 1 presents one-dimensional plots of the stimulus coordinates for Pascal (Figure 1A) and Fortran (Figure 1B) program sorts. Note that three clusters are present in both plots. One cluster contains a tight distribution of Pas1, Pas2, and Pas3 in Figure 1A and For1, For2, and For3 in Figure 1B, all members of the MGOM plan group. Another cluster contains a tight distribution of Pas4, Pas5, and Pas6 in Figure 1A and For4, For5, and For6 in Figure 1B, all members if the RCP plan group. A third cluster contains Pas7 and Pas9 in Figure 1A and For7, For8, and For9 in Figure 1B, all members of the RTRDP plan group. Note that Pas8, which should be in the RTRDP plan group, is near the members of the MGOM group. This program was the "psychology experiment" program which presents stimuli, collects reaction times, and writes means to a file. Many subjects commented that they did not really understand this program. Several subject said that they sorted it with the MGOM programs because they perform numerical calculations. Pas1 and For1, in fact, calculate means and therefore share a task with Pas8 and For8. When subjects did not understand the structure of the program, they sorted on this common task feature.

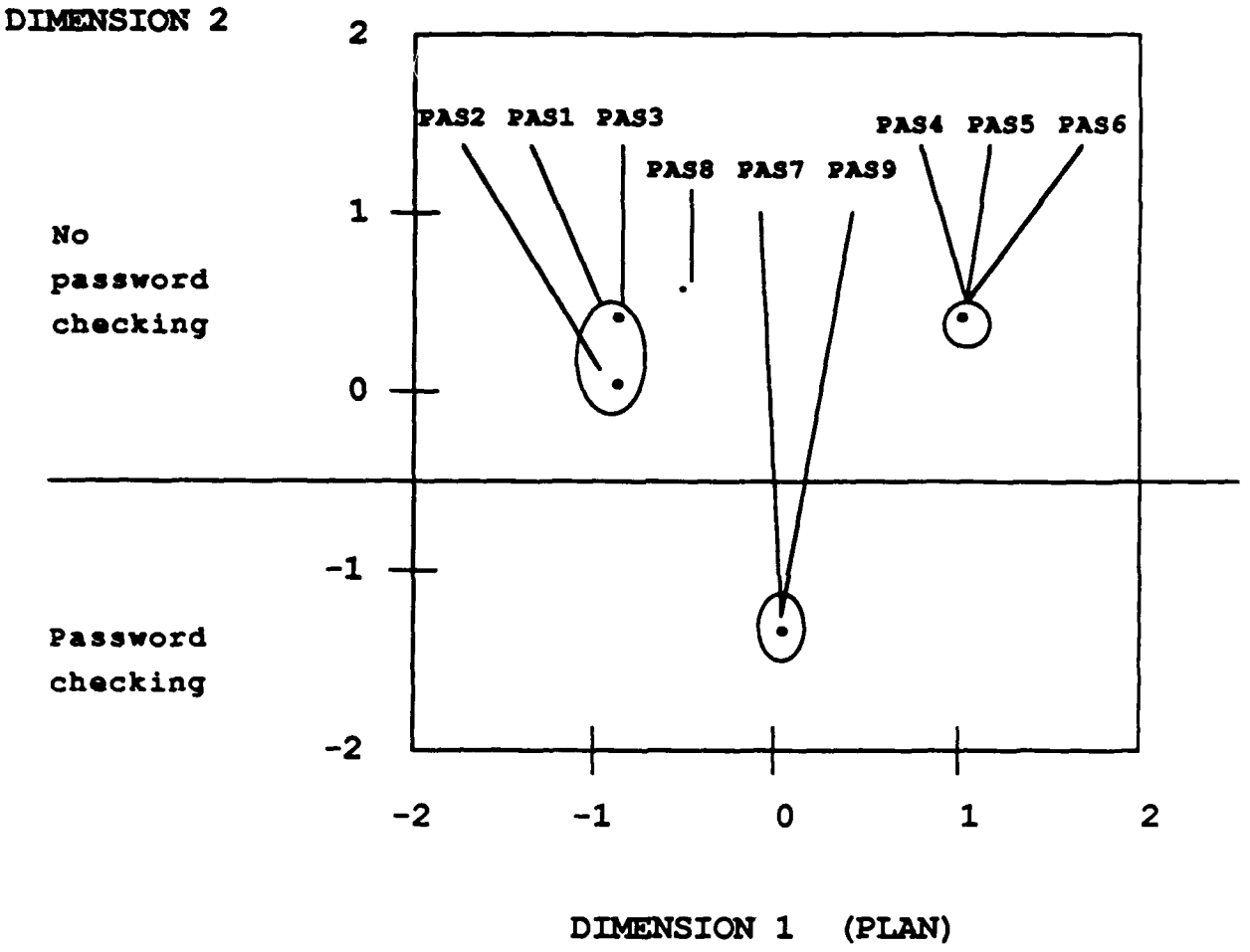


Figure 2. Plots of the stimulus coordinates for two-dimensional solutions to MDS of Pascal program sorting data.

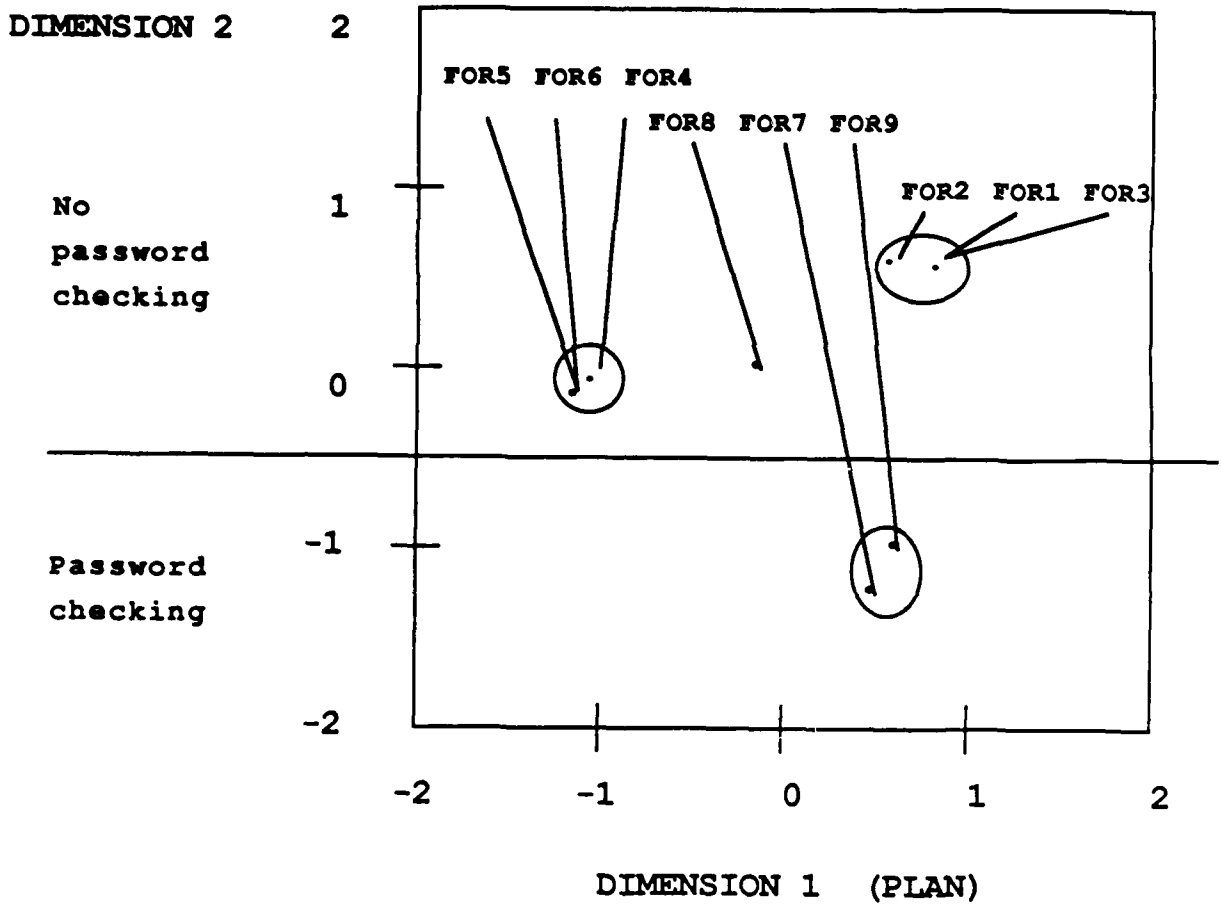
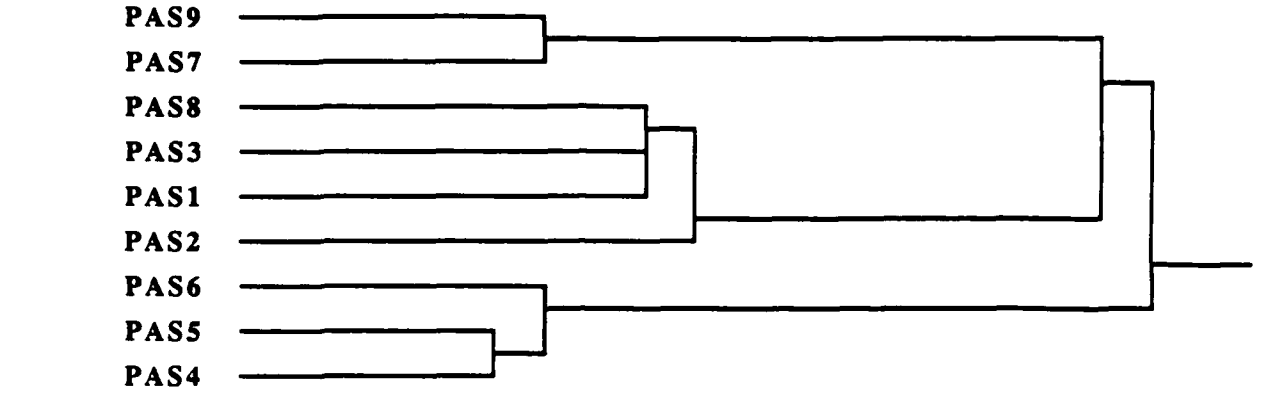


Figure 3. Plots of the stimulus coordinations for two-dimensional solutions to MDS of Fortran program sorting data.

Figures 2 and 3 present the two-dimensional plots of the stimulus coordinates for Pascal and Fortran program sorts respectively. We have circled the original plan groups in these two figures, and drawn a line to separate Pas7 and Pas9 from the other programs in Figure 2 and For7 and For9 from the other programs in Figure 3. Addition of the second dimension seems to draw these two programs away from the rest in both language groups. In looking at the descriptions that subjects gave and the contents of the programs themselves, we conclude that these programs (one is the bank teller program and the other is the login program) are distinctive since they both contain requests for passwords and require a test of the passwords before continuing.

Finally, hierarchical clustering (Johnson, 1967) is often used with sorting data to show group structure. Figure 4 shows hierarchical clustering of the sorting data for the Pascal (Figure 4a) and Fortran (Figure 4b) programs. Highly dissimilar items, indicated by infrequent sorts into the same group, cause branching high in the tree structures. More similar items cause branching lower in the trees.

For the Pascal programs (Figure 4a), the highest braching creates two groups. One group contains Pas4, Pas5, and Pas6 while the remaining programs are in the second group. Pas4, Pas5, and Pas6 are all members of the RCP plan group, programs which handle non-numerical information. The remaining programs all handle numerical information. Thus,



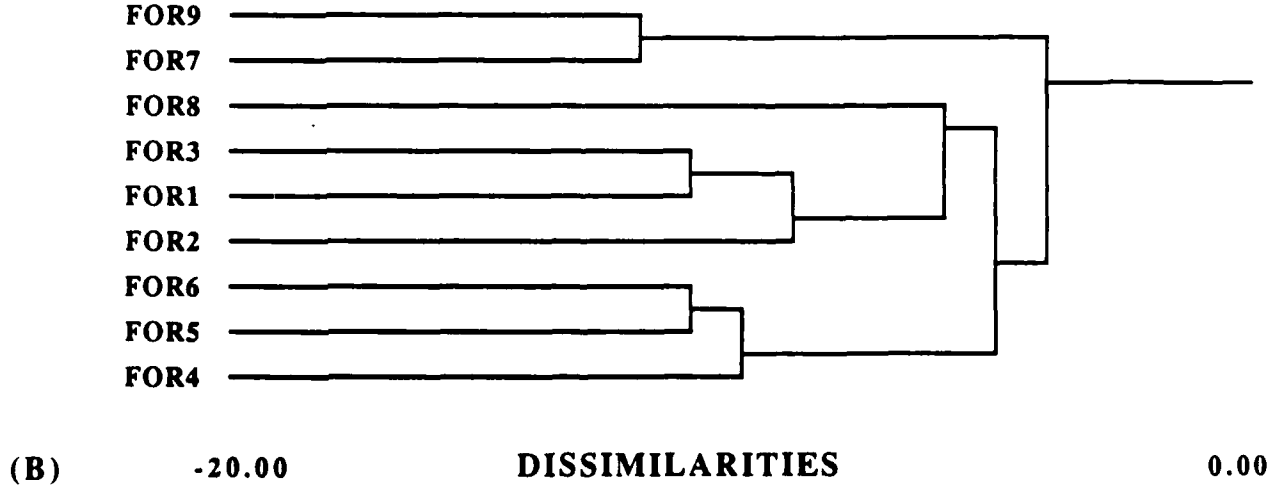


Figure 4. Hierarchical clustering of Pascal (A) and Fortran (B) program sorting data.

an important dimension for the Pascal code may be the type of data that is handled. Among the Pascal programs that handle numerical data, two more clusters emerge high in the tree. One cluster contains Pas7 and Pas9, both members of the RTRDP plan group and the two programs which check passwords. In the other cluster are Pas1, Pas2, and Pas3, all members of the MGOM plan group, and the deviant program, Pas8. The three clusters (Pas4-Pas5-Pas6, Pas5-Pas9, and Pas1-Pas2-Pas3-Pas8) do not begin to break up until much lower in the tree, suggesting that they form three important clusters which (with the exception of Pas8) correspond to the three plan groups examined in this experiment.

For the Fortran programs (Figure 4b), a similar, but not exactly equivalent picture emerges from hierarchical clustering. At the highest branching level, For7 and For9, the password programs in the RTRDP plan group, form a cluster distinct from the other programs. At the second branching level, which is still very high, For4, For5, and For6, the non-numerical RCP programs, form a cluster distinct from For1, For2, and For3, all members of the MGOM plan group, and the deviant For8. At this point, the hierarchical structure of the Pascal and Fortran programs looks very similar. Unlike Pas8 in the Pascal programs, For8 breaks away from the For1-For2-For3 cluster at a high level. This leaves three plan clusters in the Fortran data (For4-For5-For6, For7-For9, and For1-For2-For3) that remain together until low in the tree and that correspond to the

three plan groups examined in this experiment. For8 is not cleanly in any cluster.

Discussion

In this experiment subjects segmented and labelled sections of several programs and then sorted the programs into groups. The positions of segmenting lines in the programs was consistent with the predicted positions of plan chunks for both Pascal and Fortran programs written in accordance with three plans. The labels that subjects gave to these chunks were similar within plan groups, even across Pascal and Fortran language groups. A majority of subjects (60% or more) provided labels that reflected the major subgoals that plan components achieved for most of the plan chunks in the programs. Examination of the modal labels for plan chunks shows both abstract descriptions (e.g. "print results") and more task specific descriptions (e.g. "compute means"), however the bulk of the descriptions are abstract. In those cases where a majority of subjects did not provide a chunk label, we must assume that they perceived two subgoals as being combined. This occured most often when the subgoals were "Calculate" and "Print." In some cases where programs either performed a calculation and returned to a menu or quit, subjects described the "Quit" subgoal as separate from the "Calculate" subgoal.

Sorting data for both the Pascal and Fortran programs shows that they cluster into plan groups. This suggests

that subjects perceive the abstract plan structure common to all of the programs within a plan cluster and use it a basis for classifying the programs. A secondary classification criterion is based on common features. In one case, when the structure of a program was not clear to subjects, they based their sorting judgement on the fact that other programs calculated means. When a second dimension is examined, subjects appeared to be sorting on a task-specific feature, namely whether or not a program had password checking. For Pascal programs, the data type was an important feature for distinguishing programs.

We conclude from this data that programmers consider both general plan information that is common to many tasks and task-specific program constructs when comprehending code. The similarity of labels for both Pascal and Fortran data suggests that some general programming plans are common across languages. In future studies, we expect to examine this issue more carefully. Pilot data on sorting of descriptions of these Pascal and Fortran programs suggests that language information is not present in the abstract representations that are used to label the programs.

Finally, we should note that although the data, especially the sorting data, provides good evidence for abstract plan structures, we were not able to generate perfect stimuli. One of the programs was unexpectedly sorted into a different plan group, apparently on the basis of non-plan related criteria. This suggests a practical

recommendaton that all complex stimuli to be used in studies of programmers and their tasks should be empirically validated on the constructs that the experimenters feel are important. The deviant program also brings up a theoretical question. How are different types of knowledge, like general plan knowledge, task-specific knowledge, and knowledge of language constructs related and used together to reason about code? In future studies (using these materials, by the way) we will examine some of these questions.

Acknowledgements

Our thanks to Dr. John O'Hare as the principal advocate and critic of this research program and to the Office of Naval Research for their generous support. We are also grateful to David Koizumi for his assistance in the MDS and clustering analyses.

References

Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory and Cognition*, 9, 422-433.

Bonar, J., & Soloway, E. (1985). Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human-Computer Interaction*, 1, 133-161.

Ehrlich, K., & Soloway, E. (1984). An empirical investigation of tacit plan knowledge in programming. In J.C. Thomas & M.L. Schneider (Eds.), *Human factors in computer systems*. Norwood, NJ: Ablex Publishing Corporation.

Johnson, S.C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32, 241-254.

Kruskal, J.B., & Wish, M. (1978). *Multidimensional scaling*. Beverly Hills, CA: Sage Publications Inc.

Rist, R.S. (1986). Plans in programming: Definition, demonstration and development. In E. Soloway & S. Iyengar (Eds.), *Empirical Studies of Programmers*. Norwood, NJ: Ablex Publishing Corporation.

Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge, *IEEE Transactions on Software Engineering*, 5,, 595-609.

Soloway, E., Ehrlich, K., Bonar, J., & Greenspan, J. (1982). What do novices know about programming? In A. Badre & B. Schneiderman (Eds.), *Directions in human-computer interaction*. Norwood, NJ: Ablex Publishing Corporation..

Appendix A. Texts of Pascal and Fortran programs that belong to the "MGOM" plan group (6 pages).

```
PROGRAM Pas1(input,output);{ 10/03/'86 }
CONST
   No =  5;
VAR
   I,J,T,Intrvl,Temp,ERR,NoKey,Num : integer;
   Result : array [1..No] of real;
   Key : string [4];
BEGIN
   for I := 1 to 6 do writeln;
   writeln ('                          ********** MANUAL **********');
   writeln;writeln;
   writeln ('                    NUM LOCK   : press it before using');
   writeln ('                    B or b     : press it to begin processing');
   writeln ('                    BACKSPACE  : press it to backspace');
   writeln ('                    ENTER      : press it after keying');
   writeln ('                    "="        : press it for mean of single');
   writeln ('                                 group');
   writeln ('                    E or e     : press it to end one task');
   writeln ('                    C or c     : press it to clear screen');
   writeln ('                                 and continue');
   writeln ('                    Q or q     : press it to quit');
   read (Key);
   if (Key = 'B') or (Key = 'b') then
   begin
     writeln;writeln;writeln;
     write('                 *  How many groups do you want to analyze ? ');
     read(Num)
   end;
   Key:='c';
   repeat
     J:=0;
     if (Key = 'C') or (Key = 'c') then clrscr;
     for I:=1 to 6 do writeln;
     for I:= 1 to Num do Result[I]:=0;
     repeat
       I:=0;
       J:=J+1;
       Temp:=J;
       repeat
         read(Key);
         write(' ');
         I:=I+1;
         T:=I;
         val(Key,NoKey,ERR);
         Result[J]:=Result[J]+NoKey;
       until (Key = '=') or (Key = 'E') or (Key ='e');
       if (Key <> 'E') and (Key <> 'e') then
       begin
         Result[J]:=Result[J]/(T-1);
         writeln('==> MEAN : ',Result[J],'-----------(',J,')')
       end
     until (Key = 'E') or (Key = 'e');
   writeln;writeln;writeln;
   writeln('*** Q or q for QUIT, C or c for Starting again ***');
   read(Key)
   until (Key = 'Q') or (Key = 'q')
END.
```

```
C***  PROGRAM FORTRAN1 (11/12/'86)
      INTEGER I,J,T,TEMP,NUM,NOKEY
      REAL    RESULT
      CHARACTER*4 KEY
      DIMENSION RESULT(5)
      WRITE(*,10)
10    FORMAT(///////)
      WRITE(*,*)'***  MANUL  ***'
      WRITE(*,*)'NUM LOCK    : PRESS IT BEFORE USING'
      WRITE(*,*)'[B]         : TO BEGIN'
      WRITE(*,*)'BACKSPACE   : BACKSPACE'
      WRITE(*,*)'ENTER       : PRESS IT AFTER KEYING'
      WRITE(*,*)'[=]         : TO GET ONE MEAN'
      WRITE(*,*)'[E]         : END ONE TASK AND START'
      WRITE(*,*)'              AGAIN'

      WRITE(*,*)'[Q]         : QUIT'
      READ(*,20) KEY
20    FORMAT(A)
50    IF (KEY.EQ.'B') THEN
       WRITE(*,*)'* HOW MANY GROUPS YOU WANT ANALYSE ?'
       READ(*,*) NUM
      ENDIF
      IF (KEY.NE.'Q') THEN
       DO 150 I=1,NUM
       RESULT(I)=0
150    CONTINUE
       J=0
       IF (KEY.NE.'E') THEN
200     I=0
        J=J+1
300     READ(*,20) KEY
       IF((KEY.NE.'=').AND.(KEY.NE.'E').AND.(KEY.NE.'Q')) THEN
        NOKEY=ICHAR(KEY)-48
        I=I+1
        RESULT(J)=RESULT(J)+NOKEY
        GOTO 300
       ELSEIF (KEY.EQ.'=') THEN
        RESULT(J)=RESULT(J)/I
        WRITE(*,*)'==> MEAN :'
        WRITE(*,*) RESULT(J)
        GOTO 200
        ELSEIF (KEY.EQ.'E') THEN
         KEY='B'
         GOTO 50
        ENDIF
       ENDIF
      ENDIF
      STOP
      END
```

```
PROGRAM Pas2(input,output);{10/13/'86}
VAR
   I : integer;
   KeyStr : string [40];
   KeyPath : string [15];
   Key : char;
BEGIN
   for I := 1 to 6 do writeln;
   writeln ('                          ********** MANUAL **********');
   writeln;writeln;
   writeln ('                  [M]ail        : to enter massage');
   writeln ('                  [ENTER]       : press it after completing');
   writeln ('                  [BACKSPACE]   : press it to backspace');
   writeln ('                  [E]rase       : to erase massage');
   writeln ('                  [F]rom        : from whom');
   writeln ('                  [T]o          : to whom');
   writeln ('                  [N]ext        : done and for next message');
   writeln ('                  [Q]uit        : done and quit');
   read(Key);
   clrscr;
   while Key in ['M','m','E','e'] do
   begin
     for I:=1 to 10  do writeln;
     write ('          * ');
     read(Key);
     write(' : ');
     readln(KeyStr);writeln;
     write('          * ');
     read(Key);
     if not (Key in ['E','e']) then
     begin
       write(' : ');
       readln(KeyPath);
       write('          * ');
       read(Key);
       write('   : ');
       readln(KeyPath);
       write('          * ');
       read(Key);
     end;
     if (Key in ['N','n']) and (not (Key in ['e','E'])) then
     begin
       Key:='m';
       clrscr;
     end
   end;
   if Key in ['Q','q'] then
   begin
     writeln;writeln;
     writeln('          ------- All JOBS DONE, BYE ! -------')
   end
 END.
```

```
C***  PROGRAM FORTRAN2 (11/5/'86)
      INTEGER         I
      CHARACTER       KEY
      CHARACTER*15    KEYSTR,KEYPATH
      J=0
      WRITE(*,10)
10    FORMAT(///////)
      WRITE(*,*)'                    ***** MANUL ****'
      WRITE(*,20)
20    FORMAT(//)
      WRITE(*,*)'          [M]AIL         : ENTER MASSAGE'
      WRITE(*,*)'          [ENTER]        : END OF MASSAGE'
      WRITE(*,*)'          [BACKSPACE]    : BACKSPACE'
      WRITE(*,*)'          [E]RASE        : ERASE MASSAGE'
      WRITE(*,*)'          [F]ROM         : FROM WHOM'
      WRITE(*,*)'          [T]O           : TO WHOM'
      WRITE(*,*)'          [N]EXT         : DONE, FOR NEXT'
      WRITE(*,*)'          [Q]UIT         : DONE AND QUIT'
      READ(*,30) KEY
30    FORMAT(A)
100   IF ((KEY .EQ. 'M') .OR. (KEY .EQ. 'E')) THEN
       WRITE(*,*)' *'
       READ(*,30) KEYSTR
110    READ(*,30) KEY
       IF (KEY .EQ. 'E') GOTO 100
       IF (KEY .NE. 'F') THEN
        GOTO 110
       ELSE
        WRITE(*,*)' FROM : '
        READ(*,30) KEYPATH
       ENDIF
120    READ(*,30) KEY
       IF (KEY .EQ. 'E') GOTO 100
       IF (KEY .NE. 'T') THEN
        GOTO 120
       ELSE
        WRITE(*,*)' TO   :'
        READ(*,30) KEYPATH
       ENDIF
      ENDIF
      READ(*,30) KEY
      IF (KEY .EQ. 'N') THEN
       KEY='M'
       GOTO 100
      ELSEIF (KEY .EQ. 'Q') THEN
       WRITE(*,*)'!! JOBS DONE, BYE !!'
      ENDIF
      STOP
      END
```

```
PROGRAM Pas3(input,output);{09/26/'86}
VAR
   I : integer;
   NoKey,Result : real;
   Key : char;
BEGIN
   for I := 1 to 6 do writeln;
   writeln ('                          ********** MANUAL **********');
   writeln;writeln;
   writeln ('                      NUM LOCK : press it before using');
   writeln ('                      B or b   : press it to begin computing');
   writeln ('                      ENTER    : press it after keying');
   writeln ('                      C or c   : press it to clear screen');
   writeln ('                                 and continue');
   writeln ('                      Q or q   : press it to quit');
   read (Key);
   if (Key = 'B') or (Key = 'b') then Key := 'c';
   repeat
     if (Key = 'C') or (Key = 'c') then clrscr;
     Result:=0;
     Key:='+';
     repeat
       read(NoKey);
       case Key of
         '+' : Result:=Result+NoKey;
         '-' : Result:=Result-NoKey;
         '*' : Result:=Result*Nokey;
         '/' : Result:=Result/NoKey
       end;
       read(Key)
     until Key = '=';
     writeln(Result);
     read(Key)
   until (Key = 'q') or (Key = 'Q')
END.
```

```
C***  PROGRAM FORTRAN3(10/27/'86)

      INTEGER   I,NUM
      REAL      NOKEY,RESULT
      CHARACTER KEY
      NUM=10
      WRITE(*,10)
10    FORMAT(//////)
      WRITE(*,*)'                    **** MANUAL ****'
      WRITE(*,20)
20    FORMAT(//)
      WRITE(*,*)'            NUM LOCK : PRESS IT BEFORE USING'
      WRITE(*,*)'            [ B ]    : TO BEGIN COMPUTING'
      WRITE(*,*)'            ENTER    : END OF SINGLE DATA'
      WRITE(*,*)'            [ C ]    : GO ON NEXT TASK'
      WRITE(*,*)'            [ Q ]    : QUIT'
      READ(*,100) KEY
100   FORMAT(A)
200   IF ((KEY .EQ. 'B') .OR. (KEY .EQ. 'C')) THEN
       WRITE(*,*)'HERE WE BEGIN....'
       RESULT=0
       KEY='+'
300    READ(*,*) NOKEY
       IF (KEY .EQ. '+') THEN
        RESULT=RESULT+NOKEY
       ELSEIF (KEY .EQ. '-') THEN
        RESULT=RESULT-NOKEY
       ELSEIF (KEY .EQ. '*') THEN
        RESULT=RESULT*NOKEY
       ELSEIF (KEY .EQ. '/') THEN
        RESULT=RESULT/NOKEY
       ENDIF
       READ(*,100) KEY
       IF (KEY .EQ. '=') THEN
        GOTO 600
       ELSE
        GOTO 300
       ENDIF
500    CONTINUE
      ENDIF
600   WRITE(*,610) RESULT
610   FORMAT(F10.3)
650   READ(*,100) KEY
      IF ((KEY .NE. 'C') .AND. (KEY .NE. 'Q'))  GOTO 650
      IF (KEY .EQ. 'C') THEN
       GOTO 200
      ELSE
       WRITE(*,*)'                     **** THANK ****'
      ENDIF
      STOP
      END
```

Appendix B. Texts of Pascal and Fortran programs that belong to the "RCP" plan group (6 pages).

```
PROGRAM Pas4;{10/08/'86}
TYPE
    Word=string [15];
    Coursetype = record
           Course : Word;
           Crts : integer
         end;
CONST
    N1=2;
    N2=3;
VAR
    I,J,K,Ctr : integer;
    SUNY,Transf :array[1..N1] of CourseType;
    MIT : array[1..N2] of CourseType;
    Data1,Data2:text;
BEGIN
    Ctr:=0;
    assign(Data1,'data1.3');
    reset(Data1);
    for I:=1 to N1 do readln(Data1,SUNY[I].Course,SUNY[I].Crts);
    close(Data1);
    assign(Data2,'data2.3');
    reset(Data2);
    for J:=1 to N2 do readln(Data2,MIT[J].Course,MIT[J].Crts);
    close(Data2);
    for I:=1 to N1 do
    begin
      for J:=1 to N2 do
      begin
        if (SUNY[I].Crts)=MIT[J].Crts) and (SUNY[I].Course=MIT[J].Course) then
        begin
          Ctr:=Ctr+1;
          with Transf[Ctr]
          do begin
               Course:=SUNY[I].Course;
               Crts:=MIT[I].Crts
             end
        end
      end
    end;
    for I:=1 to 8 do writeln;
    writeln('                ***** COURSES CAN BE TRANSFERED *****');
    writeln;writeln;
    for I:=1 to Ctr
    do begin
        with Transf[I] do
         writeln('                    * ',I:3,'    ',Course:15,Crts:5)
       end;
    writeln;writeln('               ***** ---------- END ---------- *****')
END.
```

```
C***  PROGRAM FORTRAN4 (11/6/'86)
      INTEGER        I,J,K,C,CRT1,CRT2,CRT
      CHARACTER*15   SUNY,MIT,TRANSF
      DIMENSION      CRT1(5),CRT2(5),CRT(5)
      DIMENSION      SUNY(5),MIT(5),TRANSF(5)
      C=0
      OPEN(20,FILE='DATA1.3')
      OPEN(21,FILE='DATA2.3')
10    FORMAT(A)
20    FORMAT(I2)
      DO 100 I=1,5
      READ(20,10) SUNY(I)
      READ(20,20) CRT1(I)
100   CONTINUE
      DO 200 J=1,5
      READ(21,10) MIT(J)
      READ(21,20) CRT2(J)
200   CONTINUE
      DO 400 I=1,5
      DO 400 J=1,5
      IF (SUNY(I).NE.MIT(J)) GOTO400
      IF (CRT1(I).LT.CRT2(J)) GOTO 400
      C=C+1
      CRT(C)=CRT2(J)
      TRANSF(C)=MIT(I)
400   CONTINUE
      WRITE(*,410)
410   FORMAT(//////)
      WRITE(*,*)'**** COURSES CAN BE TRANSFERED ****'
      WRITE(*,*)'    COURSE            CREDITS'
      DO 500 I=1,C
      WRITE(*,420) TRANSF(I),CRT(I)
420   FORMAT(A,'         ',I2)
500   CONTINUE
      WRITE(*,*)'************** END ****************'
      STOP
      END
```

```
PROGRAM Pas5;{ 10/01/'86 }
TYPE
    WeekDay   = (Mon,Tue,Wen,Thu,Fri);
    Schedule = record
                 Morning   : char;
                 Noon      : char;
                 Afternoon : char
               end;
VAR
    I,OrdDay : integer;
    Lori,Bruce,ComTime : array [Mon..Fri] of Schedule;
    Day : WeekDay;
    Data1,Data2 : text;
BEGIN
    assign(Data1,'data1');
    reset(Data1);
    for Day:=Mon to Fri do
     with Lori[Day] do readln(Data1,Morning,Noon,Afternoon);
    close(Data1);
    assign(Data2,'data2');
    reset(Data2);
    for Day:=Mon to Fri do
     with Bruce[Day] do readln(Data2,Morning,Noon,Afternoon);
    close(Data2);
    for Day:=Mon to Fri
    do begin
         with Lori[Day] do
         begin
           if (Morning = Bruce[Day].Morning) and (Morning = '*')
           then ComTime[Day].Morning:='*'
           else ComTime[Day].Morning:='-';
           if (Noon = Bruce[Day].Noon) and (Noon = '*')
           then ComTime[Day].Noon:='*'
           else ComTime[Day].Noon:='-';
           if (Afternoon = Bruce[Day].Afternoon) and (Afternoon = '*')
           then ComTime[Day].Afternoon:='*'
           else ComTime[Day].Afternoon:='-'
         end
       end;
    for I:=1 to 8 do writeln;
    writeln('                          ***** THE COMMON SCHEDULE *****');
    writeln;writeln;
    writeln('                         ',' Morning       Noon       Afternoon');
    for Day:=Mon to Fri do
    begin
      OrdDay:=ord(Day)+1;
      write('                  * ',OrdDay,'      ');
      with ComTime[Day] do writeln(Morning:6,Noon:12,Afternoon:13)
    end
  END.
```

```
C***  PROGRAM FORTRAN5 (31/10/'86)

      INTEGER I,J,ORDAY,DAY
      CHARACTER LORI,BRUCE,COMTIME
      DIMENSION LORI(5,3),BRUCE(5,3),COMTIME(5,3)
      OPEN(20,FILE='DATA1')
      OPEN(21,FILE='DATA2')
10    FORMAT(A)
      DO 100 I=1,5
      DO 100 J=1,3
      READ(20,10) LORI(I,J)
      READ(21,10) BRUCE(I,J)
100   CONTINUE
      DO 200 I=1,5
      DO 200 J=1,3
      IF((LORI(I,J).EQ.BRUCE(I,J)).AND.(LORI(I,J).EQ.'*'))THEN
        COMTIME(I,J)='*'
      ELSE
        COMTIME(I,J)='-'
      ENDIF
200   CONTINUE
      WRITE(*,300)
300   FORMAT(///////)
      DO 500 I=1,5
      WRITE(*,*) I
      WRITE(*,*) COMTIME(I,1),COMTIME(I,2),COMTIME(I,3)
500   CONTINUE
      STOP
      END
```

```
PROGRAM Pas6;{09/24/'86}
TYPE
    Word=string [15];
    Psnfl=record
            Name:Word;
            Tele:Word
          end;
CONST
    N1=2;
    N2=3;
VAR
    I,J,K,Ctr:integer;
    JJ,CoList:array[1..N1] of Psnfl;
    ATT:array[1..N2] of Psnfl;
    Data1,Data2:text;
BEGIN
    Ctr:=0;
    assign(Data1,'data1');
    reset(Data1);
    for I:=1 to N1 do readln(Data1,JJ[I].Name,JJ[I].Tele);
    close(Data1);
    assign(Data2,'data2');
    reset(Data2);
    for J:=1 to N2 do readln(Data2,ATT[J].Name,ATT[J].Tele);
    close(Data2);
    for I:=1 to N1
    do begin
         for J:=1 to N2
         do begin
              if JJ[I].Name=ATT[J].Name
              then begin
                     Ctr:=Ctr+1;
                     with CoList[Ctr]
                      do begin
                           Name:=ATT[J].Name;
                           Tele:=ATT[J].Tele
                         end;
                   end
            end
       end;
    writeln('   ***** THE COMMON CUSTOMERS *****');
     for I:=1 to Ctr
    do begin
         with CoList[I] do
          writeln('   * ',I,' ',Name,Tele)
       end
END.
```

```
C***  PROGRAM FORTRAN6 (10/20/'86)

      CHARACTER*15 BELL,RCA,COLIST
      INTEGER      I,J,K,CTR
      INTEGER*4       TEL1,TEL2,COTEL
      DIMENSION BELL(2),RCA(3),COLIST(2)
      DIMENSION TEL1(2),TEL2(3),COTEL(2)
      CTR=0
      OPEN(20,FILE='DATA1.1')
      OPEN(21,FILE='DATA2.1')
      DO 100 I=1,2
      READ(20,10) BELL(I)
      READ(20,20) TEL1(I)
10    FORMAT(A)
20    FORMAT(I11)
100   CONTINUE
      DO 200 J=1,3
      READ(21,10) RCA(J)
      READ(21,20) TEL2(J)
110   FORMAT(A)
120   FORMAT(I11)
200   CONTINUE
210   DO 300 I=1,2
      DO 300 J=1,3
      IF (.NOT. (BELL(I).EQ.RCA(J))) GOTO 300
      CTR=CTR+1
      COLIST(CTR)=RCA(J)
      COTEL(CTR)=TEL2(J)
300   CONTINUE
      WRITE(*,*) '                          **** THE COMMON CUSTOMERS ****'
      DO 500 K=1,CTR
      WRITE(*,*) '      ',K,COLIST(K),' ',COTEL(K)
500   CONTINUE
      STOP
600   WRITE(*,*) 'NO SUCH FILE'
      END
```

Appendix C. Texts of Pascal and Fortran programs that belong to the "RTRDP" plan group (9 pages).

```
PROGRAM Pas7;{10/09/'86}
TYPE
    LogType = record
                  Date   : string [8];
                  IDNo   : string [9];
                PassWd   : string [7];
                Account  : real
              end;
VAR
    TempA : real;
    LogIn : LogType;
    KeyS,TempD : string [12];
    Key : char;
    I,J,NoKey : integer;
    DataC : text;
BEGIN
    assign(DataC,'datac.2');
    reset(DataC);
    with LogIn do readln(DataC,Date,IDNo,PassWd,Account);
    close(DataC);
    TempD:=LogIn.Date;
    TempA:=LogIn.Account;
    for I:=1 to 10 do writeln;
    write('                    *          Date    : ');
    readln(LogIn.Date);
    write('                    *          ID No. : ');
    readln(KeyS);
    if KeyS <> LogIn.IDNo then
    begin
      writeln(chr(007),'                    !!!! WRONG ID No., TRY AGAIN !!!!');
      write('                  *           ID No. : ');
      repeat readln(KeyS) until KeyS = LogIn.IDNo
    end;
    write('                    *        Password : ');
    readln(KeyS);
    if KeyS <> LogIn.PassWd then
    begin
      writeln(chr(007),'                      !!!! WRONG NUMBER, TRY AGAIN !!!!');
      write('                  *         Password : ');
      repeat readln(KeyS) until KeyS = LogIn.PassWd
    end;
    clrscr;
    for I:=1 to 10 do writeln;
    writeln('                       *  RECORD OF ',LogIn.IDNo,' *');
    writeln;
    writeln('                      DATE of last time : ',TempD:9);
    writeln('                      MONEY left        : ',TempA:9);
    writeln;writeln;writeln;
    writeln('                      --( HIT SPACE TO CONTINUE )--');
    repeat read(Key) until Key = ' ';
    clrscr;
    for I:=1 to 7 do writeln;
    writeln('                   OPTIONS :');
    writeln;writeln;
    writeln('                           [ P ]  :  Pascal');
    writeln('                           [ C ]  :  C language');
    writeln('                           [ Z ]  :  Zbasic');
    writeln('                           [ L ]  :  Lisp');
    writeln('                           [ Q ]  :  Quit');
    writeln;writeln;write('                             ->  ');
    repeat read(Key) until Key in ['p','c','z','l','P','F','C','Z','L'];
    clrscr;
    for I:=1 to 10 do writeln;
```

```
      'C','c' : writeln('                    **** NICE TO MEET YOU IN "C" ****');
      'Z','z' : writeln('                        **** WELCOME TO Zbasic ****');
      'L','l' : writeln('                    **** NICE TO MEET YOU IN LISP ****')
;
      'Q','q' :
    end;
    if Key in ['Q','q'] then
    else repeat read(Key) until Key in ['Q','q'];
    clrscr;
    LogIn.Account:=LogIn.Account-0.3;
    for I:=1 to 10 do writeln;
    writeln('                          Date       : ',LogIn.Date:9);
    writeln('                          MONEY left : ',LogIn.Account:9);
    writeln;writeln;
    writeln('                 -----------  GOOD BYE !  -----------')
END.
```

```
C***  PROGRAM FORTRAN7 (10/29/'86)
      INTEGER      I,J,NOKEY
      REAL         TEMPA,ACCOUNT
      CHARACTER    KEY
      CHARACTER*7 PASSWRD
      CHARACTER*8 DATE
      CHARACTER*5 IDNO,KEYS,TEMPD
      OPEN(20,FILE='DATAC.2')
      READ(20,10) DATE
      READ(20,10) IDNO
      READ(20,10) PASSWRD
      READ(20,20) ACCOUNT
10    FORMAT(A)
20    FORMAT(F7.2)
      TEMPD=DATE
      TEMPA=ACCOUNT
100   WRITE(*,*) '                *       DATE  : '
      READ(*,10) DATE
150   WRITE(*,*) '                *       ID NO : '
      READ(*,10) KEYS
      IF (KEYS .NE. IDNO) THEN
        WRITE(*,*) '           !!  WRONG, PLEASE REENTER  !!'
        GOTO 150
      ENDIF
200   WRITE(*,*) '                *     PASSWORD : '
      READ(*,10) KEYS
      IF (KEYS .NE. PASSWRD) THEN
        WRITE(*,*) '           !!  WRONG, PLEASE REENTER  !!'
        GOTO 200
      ENDIF
      WRITE(*,*) '              ==================================='
      WRITE(*,*) '              *   DTATE OF LAST ENTER : ',TEMPD
      WRITE(*,*) '              *   MONEY LEFT          : ',TEMPA
      WRITE(*,*) '              ==================================='
      WRITE(*,*) '              * OPTIONS : '
      WRITE(*,*) '                      [ 0 ]  :  ZBASIC'
      WRITE(*,*) '                      [ 1 ]  :  PASCAL'
      WRITE(*,*) '                      [ 2 ]  :  LISP'
      WRITE(*,*) '                      [ Q ]  :  QUIT'
      WRITE(*,*) '                    ==> '
      READ(*,*) NUMKEY
      IF (NUMKEY .EQ. 2) NUMKEY=-1
      IF (NUMKEY) 310,320,330
310   WRITE(*,*)'           ***   WELCOME TO LISP   ***'
      GOTO 400
320   WRITE(*,*)'           ***  WELCOME TO ZBASIC  ***'
      GOTO 400
330   WRITE(*,*)'           ***  WELCOME TO PASCAL  ***'
400   READ(*,10) KEY
      IF (KEY .NE. 'Q') GOTO 400
      ACCOUNT=ACCOUNT-3.00
      WRITE(*,*) '              ================================='
      WRITE(*,*) '               *  DATE        : ',DATE
      WRITE(*,*) '               *  MONEY LEFT  : ',ACCOUNT
      WRITE(*,*) '              ========== GOOD BYE ! =========='
```

```
STOP
END
```

```
PROGRAM Pas8;{10/03/'86}
CONST
    TrlNo = 2;
VAR
    PercntY,PercntN : real;
    Key : char;
    I,J,NoKey,SubNo,Num,ERR,Postv,Negtv : integer;
    SaveKey   : array [1..TrlNo] of char;
BEGIN
    repeat
      read(Key);
      val(Key,SubNo,ERR)
    until SubNo in [1..30];
    for I:=1 to 10 do writeln;
    writeln('    !!! PLEASE LOOK AT FIXATION POINT AFTER HITTING SPACE !!!');
    repeat read(Key) until Key = ' ';
    Num:=0;Postv:=0;Negtv:=0;
    repeat
      clrscr;
      for I:=1 to 11 do writeln;
      writeln(chr(007));
      writeln('                                          *');
      for I:=1 to 300 do for J:=1 to 300 do;
      clrscr;
      for I:=1 to 10 do writeln;
      writeln('                                        AHNX');
      writeln('                                        PRBD');
      writeln('                                        OCSU');
      for I:=1 to 5000 do;
      clrscr;
      for I:=1 to 200 do for J:=1 to 200 do;
      for I:=1 to 11 do writeln;
      write('                    ** Is "P" in second line (Y/N) ?  ');
      repeat read(Key) until Key in ['Y','N','y','n'];
      Num:=Num+1;
      SaveKey[Num]:=Key;
      if SaveKey[Num] in ['Y','y'] then Postv:=Postv+1;
      if SaveKey[Num] in ['N','n'] then Negtv:=Negtv+1
    until Num = TrlNo;
    PercntY:=Postv/TrlNo;
    PercntN:=Negtv/TrlNo;
    clrscr;
    for I:=1 to 10 do writeln;
    writeln('                          ******* RESULT *******');
    writeln;writeln;
    writeln('                  Percentage of Yes response =',PercntY:9);
    writeln('                  Percentage of No  response =',PercntN:9)
END.
```

```
C***  PROGRAM FORTRAN8 (10/23/'86)
      CHARACTER*1 KEY
      INTEGER   I,J,SUBNOV,NUM
      REAL      Y,N,POSTV,NEGTV
      WRITE(*,*) 'NO. OF SUBJECT : #'
      READ(*,5) KEY
5     FORMAT(A)
      DO 10 I=1,20
      WRITE(*,5)
10    CONTINUE
      WRITE(*,*)'!!! PLEASE FIXATE AT THE SCREEN !!!'
      POSTV=0
      NEGTV=0
      DO 400 J=1,2
      DO 20 I=1,20
      WRITE(*,*)
20    CONTINUE
      DO 25 I=1,19000
25    CONTINUE
      WRITE(*,*) '                                                    AHNX'
      WRITE(*,*) '                                                    PRBD'
      WRITE(*,*) '                                                    OCSU'
      DO 27 I=1,29000
27    CONTINUE
      DO 30 I=1,50
      WRITE(*,*)
30    CONTINUE
      WRITE(*,*) '   ?? IS [P] IN THE FIRST LINE ??'
35    READ(*,5) KEY
      IF ((KEY.NE.'Y').AND.(KEY.NE.'N')) GOTO 35
      IF (KEY.EQ.'Y') THEN
       POSTV=POSTV+1
      ELSE
       NEGTV=NEGTV+1
      ENDIF
400   CONTINUE
      Y=POSTV/2
      N=NEGTV/2
      DO 600 I=1,20
      WRITE(*,*)
600   CONTINUE
      WRITE(*,*)'            ********     RESULT     ********'
      WRITE(*,605)
605   FORMAT(//)
      WRITE(*,610) Y
610   FORMAT('               PERCENTAGE OF YES RESPONSE=',F5.2)
      WRITE(*,620) N
620   FORMAT('               PERCENTAGE  OF NO RESPONSE=',F5.2)
      STOP
      END
```

```
PROGRAM Pas9;{10/08/'86}
TYPE
    AccnType = record
                 Date : string [8];
                 Name : string [12];
               PassWd : string [7];
              Balance : real
            end;

VAR
    TempB,Amount : real;
    Account : AccnType;
    KeyS,TempD : string [12];
    Key : char;
    I,J,ERR,NoKey : integer;
    DataC : text;
BEGIN
    assign(DataC,'datac');
    reset(DataC);
    with Account do readln(DataC,Date,Name,PassWd,Balance);
    close(DataC);
    writeln(Account.Date,Account.Name,Account.PassWd,Account.Balance);
    TempB:=Account.Balance;
    TempD:=Account.Date;
    for I:=1 to 10 do writeln;
    write('                    *          Date : ');
    readln(Account.Date);
    write('                    *          Name : ');
    readln(KeyS);
    if KeyS <> Account.Name then
    begin
      writeln(chr(007),'                    !!!! WRONG NAME, TRY AGAIN !!!!');
      write('                  *           Name : ');
      repeat readln(KeyS) until KeyS = Account.Name
    end;
    write('                    *      Password : ');
    readln(KeyS);
    if KeyS <> Account.PassWd then
    begin
      writeln(chr(007),'                     !!!! WRONG NUMBER, TRY AGAIN !!!!');
      write('                  *      Password :');
      repeat readln(KeyS) until KeyS = Account.PassWd
    end;
    clrscr;
    for I:=1 to 10 do writeln;
    writeln('                      *  RECORD OF ',Account.Name,' *');
    writeln;
    writeln('                     DATE of last time : ',TempD:9);
    writeln('                     BALANCE           : ',Account.Balance:9);
    writeln;writeln;writeln;
    writeln('                     --( HIT SPACE TO CONTINUE )--');
    repeat read(Key) until Key = ' ';
    clrscr;
    for I:=1 to 7 do writeln;
    writeln('                   OPTIONS :');
    writeln;writeln;
    writeln('                          [ + ]  :  receive money');
    writeln('                          [ - ]  :  deposit');
    writeln;writeln;write('                                [ ');
    repeat read(Key) until Key in [ '+','-','/'];
    write(' ]  :  ');
    readln(Amount);
    case Key of
      '+' : Account.Balance:=Account.Balance+Amount;
```

```
    end;
    clrscr;
    for I:=1 to 10 do writeln;
    writeln('                              *  RECORD OF ',Account.Name,' *');
    writeln;
    writeln('                                 DATE    : ',Account.Date:9);
    writeln('                                 BALANCE : ',Account.Balance:9);
    writeln;writeln;
    writeln('                              ********** THANKS ******* **')
END.
```

```
C***  PROGRAM FORTRAN9 (10/22/'86)

      CHARACTER    KEY
      CHARACTER*7  PASWRD
      CHARACTER*8  DATE
      CHARACTER*12 NAME,KEYS,DTEMP
      INTEGER      I,J,NUMKEY
      REAL         BALNC,BTEMP,AMNT
      OPEN(20,FILE='DATAC')
      READ(20,10) DATE
      READ(20,10) NAME
      READ(20,10) PASWRD
      READ(20,15) BALNC
10    FORMAT(A)
15    FORMAT(F8.2)
      WRITE(*,*) DATE,NAME,BALNC
      BTEMP=BALNC
      DTEMP=DATE
      WRITE(*,*) '              *   DATE : '
      READ(*,10) DATE
20    WRITE(*,*) '              *   NAME : '
      READ(*,10) KEYS
      IF (KEYS .NE. NAME) THEN
        WRITE(*,*) '            !!  WRONG,PLEASE REENTER !!'
        GOTO 20
      ENDIF
30    WRITE(*,*) '              *   PASSWORD : '
      READ(*,10) KEYS
      IF (KEYS .NE. PASWRD) THEN
        WRITE(*,*) '            !!  WRONG,PLEASE REENTER !!'
        GOTO 30
      ENDIF
      WRITE(*,*) '           ----------------------------------'
      WRITE(*,*) '            *   DATE OF LAST ENTER : ',DTEMP
      WRITE(*,*) '            *   BALANCE            : ',BALNC
      WRITE(*,*) '           ----------------------------------'
      WRITE(*,*) '            * OPTIONS :'
      WRITE(*,*) '                    [ 1 ]   :   RECEIVE   MONEY'
      WRITE(*,*) '                    [ 0 ]   :   DEPOSIT'
      WRITE(*,*) '                  --> '
      READ(*,*) NUMKEY
100   READ(*,15) AMNT
      IF (NUMKEY) 100,200,300
200   BALNC=BALNC-AMNT
      GOTO 400
300   BALNC=BALNC+AMNT
400   WRITE(*,*) '           ----------------------------------'
      WRITE(*,*) '            * RECORD OF ',NAME,' *'
      WRITE(*,*) '              DATE      :  ',DATE
      WRITE(*,*) '              BALANCE   :  ',BALNC
      WRITE(*,*) '           ************ THANKS ************'
      STOP
      END
```

Office of Naval Research
Perceptual Science Program - Code 1142PS
Technical Reports Distribution List (4 pages)

OSD

Dr. Earl Alluisi
Office of the Deputy
Under Secretary of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D. C. 20301

DEPARTMENT OF THE NAVY

Aircrew System Branch
System Engineering Test
Directorate
U.S. Naval Test Center
Patuxent River, MD 20670

Dr. Glen Allgaier
Artificial Intelligence Branch
Code 444
Naval Electronics Ocean System
Center
San Diego, CA 92152

Mr. Philip Andrews
Naval Sea System Command
Navsea 61R2
Washington, D. C. 20362

Mr. Norm Beck
Combat Control System Department
Code 221
Naval Underwater System Center
Newport, RI 02840

Dr. Lyle D. Broemeling
Code 1111SP
Office of Naval Research
800 N. Quincy
Street Arlington, VA 22217-5000

LCDR R. Carter
Office of Chief
on Naval Operations
OP-933D3
Washington D. C. 20350

Dr. L. Chmura
Computer Science & Systems
Code 5592
Naval Research Laboratory
Washington, D. C. 20350

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 North Quincy Street
Arlington, VA 22217-5000

Commander
Naval Air System Command
Crew Station Design
NAVAIR 5313
Washinton, D. C. 20361

Dean of the Academic
Departments
U.S. Naval Academy
Annapolis, MD 21402

Director
Technical Information
Division
Code 2627
Naval Research Laboratory
Washington, DC 20375-5000

Dr. Robert A.Fleming
Human Factors Support Group
Naval Personnel Research &
Development Center
1411 South Fern Street
Arlington, VA 22217-5000

Dr. Sherman Gee
Command and Control
Technology (Code 221)
Office of Naval Technology
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Eugene E. Gloye
ONR Detachment
1030 East Green Street
Pasadena, CA 91106-2485

Mr. Jeff Grossman
Human Factors Laboratory
Code 71
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Charles Holland
Office of Naval Research
Code 1133
800 N. Quincy Street
Arlington, VA 22217-5000

Huaman Factors Branch
Code 3152
Naval Weapons Center
China Lake, CA 93555

Human Factors Department
Code N-71
Naval Training System Center
Orlando, FL 32813

Human Factors Engineering
Code 441
Naval Ocean System Center
San Diego, CA 92152

CDR Thomas Jones
Code 125
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

Mr. Todd Jones
Naval Air System Command
Code APC-2050
Washington, DC 20361-1205

Dr. Michael Letsky
Office of the Chief of Naval Operations (OP-01B7)
Washington, D. C. 20305

Lt Dennis McBride
Human Factors Branch
Pacific Missle Test Center
Point Mugu, CA 93042

LCDR Thomas Mitchell
Code 55
Naval Postgraduate School
Monterey, CA 93940

Dr. George Moeller
Human Factors Department
Naval Submarine Medical Research Lab
Naval Submarine Base
Groton, CT 06340-5900

CAPT W. Moroney
Naval Air Development Center
Code 602
Warminster, PA 18974

Dr. A. F. Norcio
Computer Science & Systems
Code 5592
Naval Research Laboratory
Washington, D.C. 20301-5000

CDR James Offutt
Office of the Secretary of Defense
Strategic Defense Initiative Organization
Washington, D.C. 20301-5000

* Perceptual Science Program
Office of Naval Research
Code 1142PS
800 N. Quincy Street
Arlinton, VA 22217-5000

Dr. Randall P. Schumaker
NRL A. I. Center
Code 7510
Naval Research Laboratory
Washington, DC 20375-5000

LCDR T. Singer
Human Factors Engineering Division
Naval Air Development Center
Wasminster, PA 18974

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Washington, D. C. 20380

Mr. James Smith
Code 121
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

Special Assistant for Marine
Corps Matters
Code OOMC
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

Mr. H. Talkington
Engineering & Computer Science
Code 09
Naval Ocean System Center
San Diego, CA 92152

DEPARTMENT OF THE ARMY

Director, Organization and
Systems Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Edgar M. Johnson
Technical Director
U.S. Army Research Institute
Alexandria, VA 22333-5600

Dr. Milton S. Katz
Director, Basic Research
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Technical Director
U.S. Army Human Engineering
Laboratory
Aberdeen Proving Ground, MD 21005

DEPARTMENT OF THE AIR FORCE

Mr. Charles Bates, Director
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB
OH 45433

Dr. Kenneth R. Boff
AF AMRL/HE
Wright-patterson AFB
OH 45433

OTHER GOVERNMENT AGENCIES

** Defense Technical
Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314

Dr. Clinton Kelly
Defense Advanced Research
Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Alan Leshner
Division of Behavior and
Neural Science
National Science Foundation
1800 G. Street, N.W.
Washington, D.C. 20550

Dr. M. C. Montemerlo
Information Science &
Human Factors, Code RC
NASA HQS
Washington, D.C. 20546

OTHER ORGANIZATIONS

Dr. Deborah Boehm-Davis
Department of Psychology
George Manson University
4400 University Drive
Fairfax, VA 22030

Dr. Stanley Deutsch
NAS-National Research Council
(COHF)
2101 Constitution Avenue, N.W.
Washington, D.C. 20418

Dr. Bruce Hamill
The Johns Hopkins University
Applied Physics Lab
Laurel, MD 20707

Dr. James H. Howard, Jr.
Department of Psychology
Catholic University
Washington, D.C. 20064

Ms. Bonnie E. John
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Thomas G. Moher
Department of Electrical
Engineering & Computer Science
University of Illinois at Chicago
P.O. Box 4348
Chicago, IL 60680

Dr. Allen Newell
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

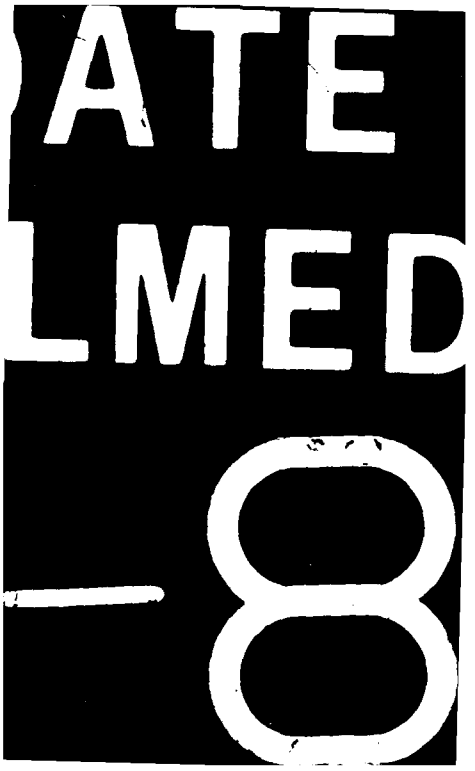
Dr. Jesse Orlansky
Institute for Defense Analysis
1801 N. Beauregard Street
Alexandria, VA 22311

Dr. Richard Pew
Bolt Bernek & Newman, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. William B. Rouse
School of Industrial and
System Engineering
Georgia Institute of
Technology
Atlanta, GA 30332

*** END OF LIST ***

* 3 copies needed
** 2 copies needed


ATE
LMED
8